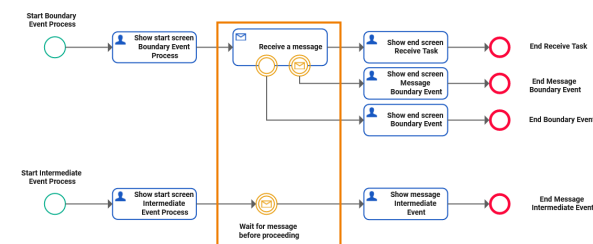






Modeling Message Reception

Via messages, you can communicate with an open process instance and provide additional data into the instance.



This can e.g. be used to synchronize the process with other applications, or to wait for additional data before continuing.

Available BPMN elements with message parameter are:

| BPMN Element | Usage | Details |
|---|---|--|
|  Receive Task | <ul style="list-style-type: none">Task that is ready to receive a message parameter.Use this task to pause process execution and wait for a message.Control the waiting time with a Timer Event. | <ul style="list-style-type: none">Receive Task |
| Message Event  | <ul style="list-style-type: none">Event that is ready to receive a message parameter.If used on a task as a boundary event, the current task is aborted, and the process flow continues from the message event on message reception.If used in the process flow as an intermediate event, process execution is paused until the message has been received. <div> A message intermediate event is similar to a receive task but cannot have a boundary event.</div> | <ul style="list-style-type: none">Message Intermediate Event |
| Message Start Event  | <ul style="list-style-type: none">Start event that is ready to receive a message parameter.Use the message start event if you need to provide data into the process at start. | <ul style="list-style-type: none">Message Start Event |

Receive Task

BPMN_Receive_Task_Example




Click the icon to download a simple example model that shows what you can do with **Receive Tasks** in **Scheer PAS Designer**.

On this Page:

- [Receive Task](#)
- [Message Event](#)
- [Message Start Event](#)
- [Messages in the Process API](#)

BPMN_Receive_Task_Example



Click the icon to download a simple example model that shows what you can do with **Receive Tasks** in **Scheer PAS Designer**.

BPMN_Event_Example



Click the icon to download a simple example model that shows what you can do with **Events** in **Scheer PAS Designer**.

Related Pages:

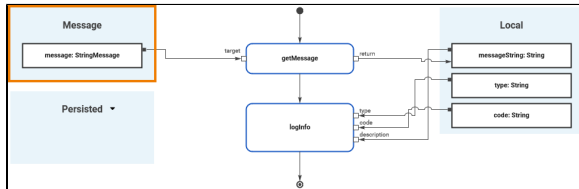
- [Modeling Process Start > Message Start Event](#)
- [Supported BPMN Elements](#)
 - [Message Event](#)
 - [Message Start Event](#)
 - [Receive Task](#)
 - [Timer Event](#)
- [Modeling Execution](#)
 - [Adding Variables](#)
 - [Persisting Data](#)
 - [Adding Operation Calls](#)
- [Technical Concepts](#)
 - [xUML Service Interface](#)
- [Testing and Integration](#)
 - [BPMN Process API Reference](#)

BPMN_Event_Example



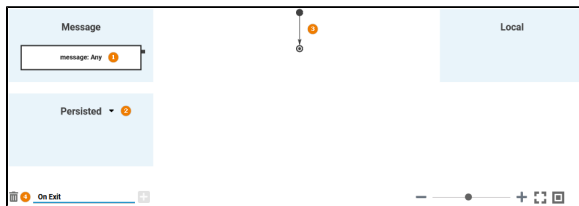
Click the icon to download a simple example model that shows what you can do with **Events** in **Scheer PAS Designer**.

A receive task is a task that is ready to receive a message parameter:



Via this message parameter, you can provide additional data to a process. The process will pause on a receive task and wait for the message reception.

A torso of the above displayed execution diagram (**On Exit 4**) is already created when adding a receive task:



- Parameter **message** has type **Any** in this torso (1). You need to update this with the actual type of the message.
Refer to [Adding Variables](#) for more information on how to do that.
- If you want to access data from the message in the process later on, you need to at least persist the message parameter (2).
Refer to [Persisting Data](#) for more information on how to do that.
- You can add more executional parts to this execution diagram (3, like shown further above).
Refer to [Adding Operation Calls](#) for more information on how to do that.

If you remove the execution diagram (4), the message parameter is dropped.



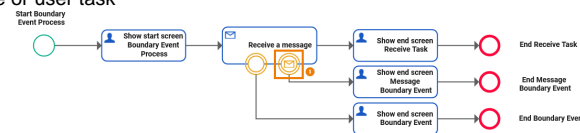
Deleting the execution model would only make sense in very special cases, e.g. if you do not need the parameter anymore but do not want to change the interface of the service. If you don't need the message parameter, use a service task instead.

Message Event

MultiExcerpt named BPMN_Intermediate_Event_Example was not found -- Please check the page name and MultiExcerpt name used in the MultiExcerpt-Include macro

A message event is similar to a receive task but cannot have a boundary event. It can be used in two ways:

1. on the boundary of a receive or user task



2. in the process sequence flow



- If used as a message boundary event, the reception of the message ends the task the boundary event is used on.
 - If the message arrives, the process ends the receive or user task, and continues with the sequence flow on the message event.
 - If the message event does not arrive, the process stays in the task the boundary event is used on.
- If used in the sequence flow of the process, the process pauses and waits for the message before continuing.

A message event can have execution: It is ready to receive a message parameter. To the execution of a message event, the same applies as for [receive tasks \(see above\)](#).

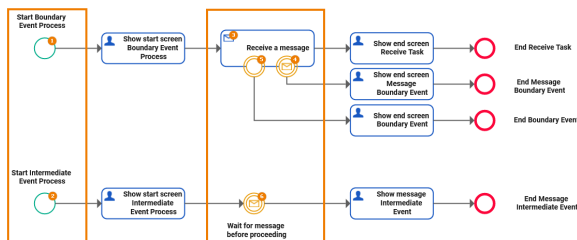
Message Start Event

MultiExcerpt named BPMN_Intermediate_Event_Example was not found -- Please check the page name and MultiExcerpt name used in the MultiExcerpt-Include macro

A message start event is a start event that is ready to receive a message. You can find more details on the message start event on [Message Start Event](#), and all about using message start events on [Modeling Process Start > Message Start Event](#).

Messages in the Process API

For message receiving process elements, the API of the generated xUML service has POST operations that can be used to send the message to the process as a body parameter. For the **BPMN_Event_Example**, that would be six in total:



- two POST operations, one for each start event (1,2)
- four POST operations, one for each message related task or event
 - [Receive_a_message](#) (receive task, 3)
 - [Message_Event](#) (message boundary event, 4)
 - [Plain_Event](#) (boundary event, no message, 5)
 - [Wait_for_message_before_proceeding](#) (message intermediate event, 6)

Each of the POST operations get the process instance **id** via the path, and the related message parameter in the request **body**.

Refer to [xUML Service Interface](#) for more information on the xUML service API in general, and to [Testing and Integration](#) for details on how to access the API and a list of the available operations.